EasyTicket: A Ticket Routing Recommendation Engine for Enterprise Problem Resolution

Qihong Shao, Yi Chen
Arizona State University
{qihong.shao, yi}@asu.edu

Shu Tao, Xifeng Yan, Nikos Anerousis
IBM T. J. Watson Research Center
{shutao, xifengyan, nikos}@us.ibm.com

ABSTRACT

Managing problem tickets is a key issue in IT service industry. A large service provider may handle thousands of problem tickets from its customers on a daily basis. The efficiency of processing these tickets highly depends on ticket routing—transferring problem tickets among expert groups in search of the right resolver to the ticket. Despite that many ticket management systems are available, ticket routing in these systems is still manually operated by support personnel. In this demo, we introduce EasyTicket, a ticket routing recommendation engine that helps automate this process. By mining ticket history data, we model an enterprise social network that represents the functional relationships among various expert groups in ticket routing. Based on this network, our system then provides routing recommendations to new tickets. Our experimental studies on 1.4 million real-world problem tickets show that on average, EasyTicket can improve the efficiency of ticket routing by 35%.

1. INTRODUCTION

Motivation: Managing problem tickets is a key issue in IT service industry. Every day, the help desk or call center of service providers, e.g., IBM, AT&T, and VISA, may receive thousands of phone calls and emails from their customers who are seeking technical supports. The reported problems range widely from login failure, application crash, to broken transactions. The efficiency of resolving these problems is a critical measurement to the productivity of a service provider.

The process of problem resolution is reflected by the life cycle of problem tickets. A ticket is opened as soon as a problem is reported, and routed among various expert groups until it reaches a resolver group that can solve the problem and close the ticket. Table 1 shows a lifecycle of a sample ticket routed among multiple groups before it was solved. As one can see, the efficiency of problem resolution hinges critically on that of ticket routing.

ID	Time	Entry
28120	2007-05-14	New Ticket: DB2 login failure
28120	2007-05-14	Transferred to Group SMRDX
28120	2007-05-14	Contacted Mary...
28120	2007-05-14	Transferred to Group SSDSISAP
28120	2007-05-14	Status updated ...
28120	2007-05-15	Transferred to Group ASWWCUST
28120	2007-05-15	Web service checking
28120	2007-05-18	Could not solve the problem.
...	...	...
28120	2007-05-18	Transferred to Group SSSAPHWOA
28120	2007-05-22	Resolved

Table 1: A sample ticket lifecycle

Today, many ticket management systems provide a collaborative platform where users and support personnel can interact in real time to report, diagnose and resolve tickets. However, ticket routing in these systems are often still manually decided by the support personnel. Making wise decisions in problem routing is extremely challenging in practice due to two factors: (1) the great diversity of reported problems and (2) the large number of expert groups to choose. For example, a set of ticket data obtained from IBM has 553 problem categories and involves more than 50 groups in resolving problems in each category. It is not uncommon that due to the limited knowledge and experience, a ticket is mistakenly transferred to a group that cannot solve the problem, which leads to a long routing sequence and excessive delay.

In this demo, we introduce EasyTicket, a recommendation engine that improves the efficiency of ticket routing (hence problem resolution) by mining the historical ticket resolution sequences.

Typically a problem ticket contains two types of information: (1) *ticket content* that includes problem description and diagnostic data, (2) *routing sequence* that shows how it was routed between different groups. In the sample ticket shown in Table 1, the entries compose the ticket content, while the extracted group names ⟨`SMRDX`, `SSDSISAP`, `ASWWCUST`, `SSSAPHWOA`⟩ form its routing sequence. Our study here focuses only on *routing sequences*. As shown in this demo, mining the routing sequences alone can significantly improve the overall efficiency of ticket routing.

Problem Definition: We now briefly describe the data model and define the problem of ticket routing recommendation. A problem ticket can be represented by a tuple with two components, $(\tau, G_{(k)})$, where τ is the ticket content and $G_{(k)}$ is the routing sequence. Let $\mathcal{G} = \{g_1, g_2, \ldots, g_n\}$ be the

PVLDB '08, August 23-28, 2008, Auckland, New Zealand

set of all expert groups. The routing sequence of a ticket can be written as $G_{(k)} = \langle g_{(1)}, g_{(2)}, \ldots, g_{(k)} \rangle$ ($g_{(i)} \in \mathcal{G}$), in which a ticket is first issued to $g_{(1)}$, then transferred in the order of $g_{(2)}, g_{(3)}, \ldots, g_{(k)}$. A *step* in $G_{(k)}$ is a ticket transfer from one group to another. A ticket $(\tau, G_{(k)})$ is open if none of the groups in $G_{(k)}$ can resolve it. Correspondingly, a ticket is closed if the last group in $G_{(k)}$, i.e., $g_{(k)}$, solved the problem, and in this case, the routing sequence is called a *resolution sequence*.

The problem of ticket routing recommendation is specified as follows:

Input: A database of historical ticket resolution sequences D_S and an open problem ticket (τ_j, G_j).

Output: Rankings of g_i ($g_i \in \mathcal{G}$) as the next group that the open ticket should be transferred to.

Objective: To improve the overall efficiency of problem resolution, measured by the Mean number of Steps To Resolve (MSTR) a set of m open tickets:

$$T = \frac{\sum_{j=1}^{m} |G_j|}{m}. \quad (1)$$

Approach: With an in-depth analysis, we find that the majority of local ticket transfer decisions were logically correct. Intuitively, the group holding a ticket generally can make a correct decision on selecting the next group to be the most likely problem resolver, based on its knowledge of the skills and expertise of the related groups. Such relations between different groups form a "social network" that often exists in enterprise problem resolution [3, 1]. However, a few local mis-routing decisions often result in long resolution sequences.

To avoid such mis-routings, we build a model that captures the social network between the expert groups, by mining the local ticket transfer decisions recorded in the resolved tickets in each problem category. The obtained model is probabilistic, because the connections between a group and the other are uncertain, even with respect to solving a specific type of problems. Using this probabilistic model, we can guide future ticket routing toward the group that is most likely to resolve the problem.

Our method is based on Markov modeling, which is a natural choice to capture the likelihood of a group to be a transfer target given the previous groups [2]. Let each Markov state represents a group, the transition probabilities between these states capture the local decisions, i.e., the likelihood of a group to be a transfer target, given the previous groups that have processed the ticket. There are several challenges in applying Markov model in this problem. What kind of group transitions should be captured by the model? What should be the optimal order of the derived Markov model? How should the model be used to guide ticket transfer?

In our study, we address the above challenges and develop a system that generates ticket routing recommendations. Following its recommendations, the MSTR of new tickets can be greatly reduced, compared to that based on human decisions. More details can be found at [4].

2. SYSTEM ARCHITECTURE

The architecture of the EasyTicket system is presented in Fig. 1. *Historical Ticket Database* collects and organizes historical tickets from different ticket management systems. Then the *Sequence Extractor* extracts ticket resolution sequences from this database in building a *Sequence Database.*

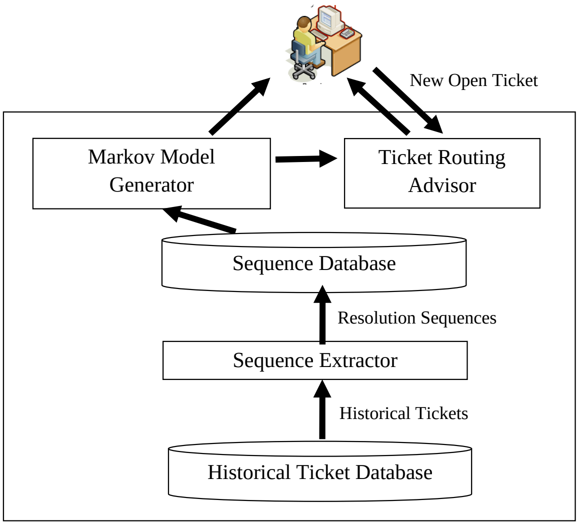


Figure 1: System architecture for EasyTicket

Markov Model Generator builds the Markov model that captures the social network that reflects the relationships between different expert groups. Finally, for an open ticket, *Ticket Routing Advisor* provides routing recommendations based on the developed model and the current routing sequence of the ticket.

2.1 Markov Modeling

We first extract routing sequences from problem tickets, e.g., sequence pattern ⟨`SMRDX`, `SSDSISAP`, `ASWWCUST`, `SSSAPHWOA`⟩ from the sample ticket in Table 1. Since the problems of different categories have different characteristics, we build up a hybrid Markov model for historical resolution sequences in each individual problem category, where the category of a problem ticket is typically assigned by the help desk that receives the problem report.

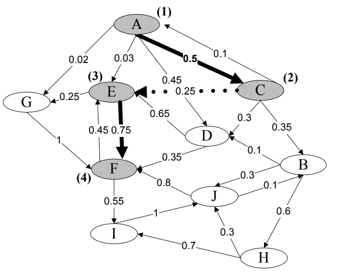
Let us use $S_{(k)}$ to denote the set of groups in $G_{(k)}$, i.e. $S_{(k)} = \{g_{(1)}, g_{(2)}, ..., g_{(k)}\}$. The number of instances with a set of group transfers $S_{(k)}$ is denoted as $N(S_{(k)})$; and the number of instances of transferring a ticket to group g_i after being processed by $S_{(k)}$ is denoted as $N(g_i, S_{(k)})$. We can estimate $P(g_i|S_{(k)})$ by

$$P(g_i|S_{(k)}) = \begin{cases} N(g_i, S_{(k)})/N(S_{(k)}) & \text{if } N(S_{(k)}) > 0, \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Our model is built based on all intermediate transfer steps in ticket resolution sequence. We define $J(\tau, g_i) = 1$ if the problem τ should go through group g_i (i.e., the problem can be either solved or correctly routed by g_i), and 0 otherwise. Then, Eq. (2) is evaluated as

$$P(g_i|S_{(k)}) = P(J(\tau, g_i)|S_{(k)}). \quad (3)$$

The order of a Markov model determines how many past states are considered to *predict* the future state of the process. To determine the "optimal" order of a Markov model, we consider the conditional entropy of the training data and evaluate the entropy of the next group g conditioned on a



(a) VMS Routing Steps

$S_{(2)}$	Next group g_i	$P(g_i \mid S_{(2)})$
{A,C}	E	0.6
{A,C}	D	0.2
{A,C}	B	0.1
{A,C}	G	0.1
{E,C}	F	0.6
{E,C}	G	0.4
…	…	…

(b) 2nd order probability table

Figure 2: Example of VMS algorithm

given set of past groups $S_{(k)}$, which is denoted as $H(g|S_{(k)})$:

$$H(g|S_{(k)}) = -\sum_{S_{(k)} \in \mathcal{G}^k} P(S_{(k)}) \sum_{g \in \mathcal{G}} P(g|S_{(k)}) \log P(g|S_{(k)}) \tag{4}$$

Specifically, users can set a threshold θ to determine the optimal order k, where k is set as the smallest value that satisfies

$$H(g|S_{(k)}) - H(g|S_{(k+1)}) < \theta. \tag{5}$$

2.2 Routing Recommendation: VMS Algorithm

Our Markov model captures the likelihood that a ticket would be transferred to a group, given the past group transfer information. The next issue is how to use it to make effective ticket routing recommendations, so that a new ticket can be transferred to its resolver group as quickly as possible. Note that the right resolver group for a ticket is unknown at the beginning of ticket routing. What we know is the initial group that a problem ticket was assigned by the help desk according to the reported problem symptoms.

We introduce a heuristic search algorithm, called *Variable-order Multiple active state Search* (VMS). It maintains a visited group set, L_v, and a candidate group set, L_c, which consists of the unvisited neighbors of all groups in L_v. It selects a group from L_c in each iteration and expands L_v, until the resolver group is found. VMS first checks all available transfer probabilities $P(g|S_{(k)}), S_{(k)} \subseteq L_v$, for all group sets visited in the past. Then, it selects the next group g^* that maximizes the transfer probability from $S_{(k)}$,

$$g^* = \text{argmax}_g \; P(g|S_{(k)}), \forall g \in L_c, S_{(k)} \subseteq L_v. \tag{6}$$

Example 1: Fig. 2 (a) shows a sample Markov model, where the value on each edge is the 1st-order transfer probability between groups, estimated by Eq. (2), and the 2nd-order transition probabilities are listed in Fig. 2 (b). Assume that we decide the optimal order $k = 2$ using Eq. (5).

Suppose an incoming ticket is initially assigned to group A and the expected resolver group is F. The VMS algorithm works as follows. Starting from the initial $L_v = \{A\}$, since only the 1st-order model is applicable at this time, the algorithm transfers the ticket to group C and updates L_v to $\{A, C\}$. Now the algorithm has the choices of using either 1st- or 2nd-order Markov model. We find that the highest conditional probability in the 2nd-order model, $P(E|A, C) = 0.6$, is greater than that in the 1st-order model, $P(D|A) = 0.45$. So the algorithm chooses E as the next group, since the 2nd-order model predicts with higher confidence. In Fig. 2, we use dashed thick line to represent this transfer. From group E, $P(F|E) = 0.75$ is the highest conditional probability for all candidate groups in L_c, even compared to 2nd-order probabilities, hence F is selected next. Thus, the VMS algorithm finally reaches the resolver group F in 4 steps: $A \rightarrow C \rightarrow E \rightarrow F$. ■

3. EVALUATION AND DEMONSTRATION

Evaluation: Toward our goal of improving problem resolution, we built the EasyTicket prototype system using Java and DB2.

We evaluated our system on a set of 1.4 million tickets in 553 problem categories, collected from IBM's problem management system between Jan 1, 2006 and Dec 31, 2006. The dataset is partitioned into training and testing sets. We first built the Markov models based on the training set. Then for each ticket in the testing set, given its initial group assignment, we applied the proposed VMS algorithm to generate routing recommendations. The effectiveness of our system is evaluated by comparing the resolution sequences resulted from EasyTicket recommendations with the ones by human decisions.

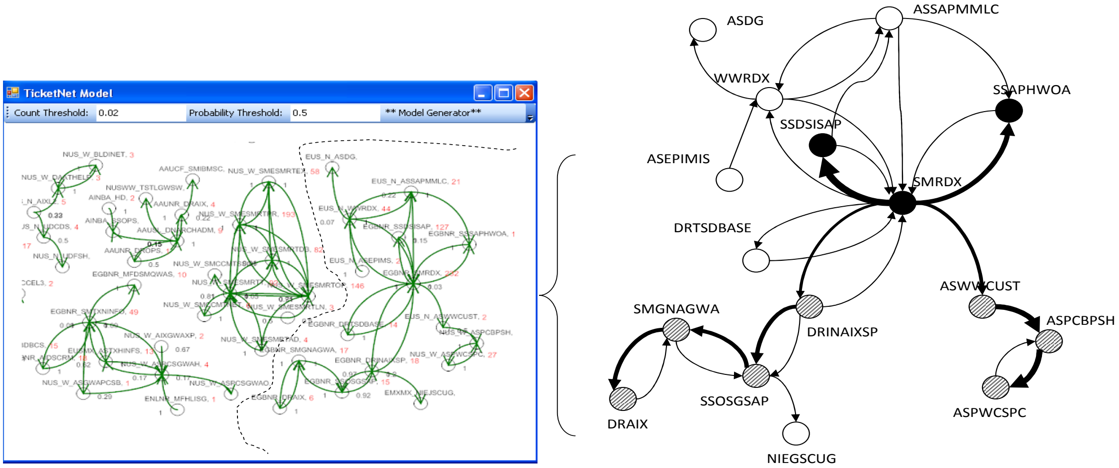
Table 2 shows the MSTR for testing tickets in five problem categories, as well as the overall MSTR for all problem categories. The fourth column is the improvement gained by using EasyTicket. It clearly shows the effectiveness of EasyTicket in improving the efficiency of ticket routing: it reduces the overall MSTR from 3.94 (based on human decisions) to 2.58. In particular, EasyTicket can effectively shorten the resolution sequence of those complex tickets that are likely to be mis-routed and cause most customer dissatisfaction. We also test the sensitivity of our approach with respect to the size of training set, time-variability of tickets, and diversity of problem categories. The results show that our solution consistently achieves good performance.

EasyTicket is not only effective, but efficient. For each problem category, it builds the proposed Markov model in less than 1 second, and the time for generating ticket routing recommendations is in average less than 1 ms per ticket.

Category	Human	ET	% reduction
ADSM	5.37	3.23	37.99%
AIX	4.89	2.78	43.15%
BIOS	4.49	2.94	34.52%
DB2	4.78	2.57	46.23%
WINDOWS	3.93	2.86	27.23%
All 553 Categories	3.94	2.58	34.52%

Table 2: MSTR for different problem categories: human decisions vs. EasyTicket(ET) recommendations.

Demonstration: To use EasyTicket, the user first specifies a problem category (e.g., DB2, Windows), and the history period to build the Markov model. Based on the specification, our system will automatically extract the resolution sequences from the history data, build and visualize the model. For example, Fig.3 (a) shows a screen shot of EasyTicket, which is a graphic presentation of the derived Markov model for DB2 problems, where each node represents an expert group and an edge represents a possible transfer between



Count Threshold: 0.02

Probability Threshold: 0.5

(a)Markov Model (b)Case study

Figure 3: User interface of EasyTicket

the groups. As discussed earlier, this model represents the enterprise social network that captures the functional relationships among expert groups.

For problem ticket resolution, EasyTicket supports two modes: *recommendation* and *simulation*. For the former, it provides the user a ranked list of recommended groups to route the testing ticket. The user can either choose one of the recommended groups, or check all relevant groups involved in the corresponding problem category and select the next group based on her own judgement. For the latter, the user specifies the initial group of a testing ticket and asks EasyTicket to automatically route the ticket until it reaches the resolver group.

To compare EasyTicket and human decision-based ticket routing, we provide a *routing comparison* module. For a user specified problem ticket, our system will graphically highlight both the routing made by human decisions and the one recommended by EasyTicket. For instance, the comparison for a sample problem ticket is illustrated in Fig. 3(b). Nine different expert groups (nodes marked black and gray in the figure) were involved in resolving this sample ticket based on human decisions. Using EasyTicket, only three groups (nodes marked black) are involved to resolve the ticket: ⟨`SMRDX`, `SSDSISAP`, `SSSAPHWOA`⟩. We can see that the inefficiency of human decision-based routing is due to two wrong local decisions that transferred the ticket from `SMRDX` to `DRINAIXSP` and from `SMRDX` to `ASWWCUST`.

Why this work is interesting to the database community? This work introduces a new problem domain to the database community—enterprise problem management. With the IT industry transforming into a service-oriented industry, problem management has played an important role in driving its growth. In problem management, enterprises often need to develop applications to effectively manage large sets of problem ticket data, and more importantly, derive business intelligence by mining the data. These applications need to not only be able to keep up with the increasingly large data volume, but also be computationally efficient for on-line usage in many cases.

The ticket routing recommendation engine introduced in this work is an example of such applications. Following this work, there are several potential extensions that are related to data management and data mining research. For instance, to fully exploit the information recorded in ticket content, text mining techniques may be applied to enhance our social network model and further improve the efficiency of ticket routing. Furthermore, the social network model built in this study can not only be used for ticket routing, but for discovering organizational issues, performance benchmarking, etc., in the effort of improving enterprise problem management in general.

The ticket routing problem is also related to web usage mining, where association rules and sequential patterns are applied to ease web access and improve the website design (our Markov model shares the similar idea with association rules). Unfortunately, in web usage applications, the model is hard to evaluate, while in ticket resolution sequence mining, the model we built can be evaluated accurately. We believe resolution sequence data provides us a good platform to experiment and demonstrate the usage and the effectiveness of sequence mining.

4. ACKNOWLEDGMENTS

This research was supported in part by NSF grant IIS-0740129 and IIS-0612273. We would like to thank Weiwei Xu and Yichuan Cai for their valuable input to the system.

5. REFERENCES

[1] Christopher S. Campbell, Paul P. Maglio, Alex Cozzi, and Byron Dom. Expertise identification using email communications. In *CIKM '03*.

[2] W. Gaaloul, S. Bhiri, and C. Godart. Discovering workflow transactional behavior from event-based log. In *Proc 12th Int'l Conf. CoopIS*, 2004.

[3] Henry Kautz, Bart Selman, and Mehul Shah. Referral web: combining social networks and collaborative filtering. 1997.

[4] Qihong Shao, Yi Chen, Shu Tao, Xifeng Yan, and Nikos Anerousis. Efficient ticket routing by resolution sequence mining. In *Proc. ACM SIGKDD*, 2008.